Terms used **garbage collection cache heap allocation evict siz** Found **47** of **182,223**

Sort results by: relevance | Display results: expanded form | Save results to a Binder | Search Tips | Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 47 Result page: **1** 2 3 next

Relevance scale

1 New garbage collection algorithms and strategies: Automatic heap sizing: taking real memory into account

Ting Yang, Matthew Hertz, Emery D. Berger, Scott F. Kaplan, J. Eliot B. Moss
October 2004 **Proceedings of the 4th international symposium on Memory management**
Publisher: ACM Press
Full text available: pdf(879.86 KB) Additional Information: full citation, abstract, references, index terms

Heap size has a huge impact on the performance of garbage collected applications. A heap that barely meets the application's needs causes excessive GC overhead, while a heap that exceeds physical memory induces paging. Choosing the best heap size a priori is impossible in multiprogrammed environments, where physical memory allocations to processes change constantly. We present an automatic heap-sizing algorithm applicable to different garbage collectors with only modest changes ...

Keywords: garbage collection, paging, virtual memory

2 Caching considerations for generational garbage collection

Paul R. Wilson, Michael S. Lam, Thomas G. Moher
January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming LFP '92**, Volume V Issue 1
Publisher: ACM Press
Full text available: pdf(1.09 MB) Additional Information: full citation, references, citings, index terms

3 Memory system performance of programs with intensive heap allocation

Amer Diwan, David Tarditi, Eliot Moss
August 1995 **ACM Transactions on Computer Systems (TOCS)**, Volume 13 Issue 3
Publisher: ACM Press
Full text available: pdf(2.10 MB) Additional Information: full citation, abstract, references, citings, index terms

Heap allocation with copying garbage collection is a general storage management technique for programming languages. It is believed to have poor memory system performance. To investigate this, we conducted an in-depth study of the memory system performance of heap allocation for memory systems found on many machines. We studied the performance of mostly functional Standard ML programs which made heavy use of

heap allocation. We found that most machines support heap allocation poorly. Howeve ...

Keywords: automatic storage reclamation, copying garbage collection, garbage collection, generational garbage collection, heap allocation, page mode, subblock placement, write through, write-back, write-buffer, write-miss policy, write-policy

4 Garbage collection without paging

Matthew Hertz, Yi Feng, Emery D. Berger

June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6

Publisher: ACM Press

Full text available: pdf(231.14 KB) Additional Information: full citation, abstract, references, citings, index terms

Garbage collection offers numerous software engineering advantages, but interacts poorly with virtual memory managers. Existing garbage collectors require far more pages than the application's working set and touch pages without regard to which ones are in memory, especially during full-heap garbage collection. The resulting paging can cause throughput to plummet and pause times to spike up to seconds or even minutes. We present a garbage collector that avoids paging. This *bookmarking collect ...*

***Keywords**: bookmarking collection, garbage collection, generational collection, memory pressure, paging, virtual memory*

5 Myths and realities: the performance impact of garbage collection

Stephen M. Blackburn, Perry Cheng, Kathryn S. McKinley

June 2004 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the joint international conference on Measurement and modeling of computer systems SIGMETRICS '04/Performance '04**, Volume 32 Issue 1

Publisher: ACM Press

Full text available: pdf(305.06 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper explores and quantifies garbage collection behavior for three whole heap collectors and generational counterparts: *copying semi-space, mark-sweep,* and *reference counting*, the canonical algorithms from which essentially all other collection algorithms are derived. Efficient implementations in MMTk, a Java memory management toolkit, in IBM's Jikes RVM share all common mechanisms to provide a clean experimental platform. Instrumentation separates collector and program behav ...

Keywords: generational, java, mark-sweep, reference counting, semi-space

6 Memory subsystem performance of programs using copying garbage collection

Amer Diwan, David Tarditi, Eliot Moss

February 1994 **Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Heap allocation with copying garbage collection is believed to have poor memory subsystem performance. We conducted a study of the memory subsystem performance of heap allocation for memory subsystems found on many machines. We found that many machines support heap allocation poorly. However, with the appropriate memory

subsystem organization, heap allocation can have good memory subsystem performance.

7 Adaptive techniques: Optimistic stack allocation for java-like languages

Erik Corry
June 2006 **Proceedings of the 2006 international symposium on Memory management ISMM '06**
Publisher: ACM Press
Full text available: pdf(155.23 KB) Additional Information: full citation, abstract, references, index terms

Stack allocation of objects offers more efficient use of cache memories on modern computers, but finding objects that can be safely stack allocated is difficult, as interprocedural escape analysis is imprecise in the presence of virtual method dispatch and dynamic class loading. We present a new technique for doing optimistic stack allocation of objects. Our technique does not require interprocedural analysis and is effective in the presence of dynamic class loading, reflection and exception han ...

Keywords: Java, garbage collection, stack allocation

8 Cache performance of fast-allocating programs

Marcelo J. R. Gonçalves, Andrew W. Appel
October 1995 **Proceedings of the seventh international conference on Functional programming languages and computer architecture**
Publisher: ACM Press
Full text available: pdf(1.47 MB) Additional Information: full citation, references, citings, index terms

9 CONS should not CONS its arguments, or, a lazy alloc is a smart alloc

Henry G. Baker
March 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 3
Publisher: ACM Press
Full text available: pdf(1.52 MB) Additional Information: full citation, abstract, index terms

Lazy allocation is a model for allocating objects on the execution stack of a high-level language which does not create dangling references. Our model provides safe transportation into the heap for objects that may survive the deallocation of the surrounding stack frame. Space for objects that do not survive the deallocation of the surrounding stack frame is reclaimed without additional effort when the stack is popped. Lazy allocation thus performs a first-level garbage collection, and if ...

10 Quantifying the performance of garbage collection vs. explicit memory management

Matthew Hertz, Emery D. Berger
October 2005 **ACM SIGPLAN Notices , Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '05**, Volume 40 Issue 10
Publisher: ACM Press
Full text available: pdf(1.51 MB) Additional Information: full citation, abstract, references, citings, index terms

Garbage collection yields numerous software engineering benefits, but its quantitative impact on performance remains elusive. One can compare the cost of *conservative* garbage collection to explicit memory management in C/C++ programs by linking in an appropriate collector. This kind of direct comparison is not possible for languages designed for garbage collection (e.g., Java), because programs in these languages naturally do not contain calls to free. Thus, the actual gap between the tim ...

Keywords: explicit memory management, garbage collection, oracular memory management, paging, performance analysis, throughput, time-space tradeoff

11 Partitioned garbage collection of a large object store

Umesh Maheshwari, Barbara Liskov

June 1997 **ACM SIGMOD Record , Proceedings of the 1997 ACM SIGMOD international conference on Management of data SIGMOD '97**, Volume 26 Issue 2

Publisher: ACM Press

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

We present new techniques for efficient garbage collection in a large persistent object store. The store is divided into partitions that are collected independently using information about inter-partition references. This information is maintained on disk so that it can be recovered after a crash. We use new techniques to organize and update this information while avoiding disk accesses. We also present a new global marking scheme to collect cyclic garbage across partitions. Global marking ...

Keywords: cyclic garbage, garbage collection, object database, partitions

12 Reducing garbage collector cache misses

Hans-J. Boehm

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(774.19 KB) Additional Information: full citation, abstract, citings, index terms

Cache misses are currently a major factor in the cost of garbage collection, and we expect them to dominate in the future. Traditional garbage collection algorithms exhibit relatively little temporal locality; each live object in the heap is likely to be touched exactly once during each garbage collection. We measure two techniques for dealing with this issue: prefetch-on-grey, and lazy sweeping. The first of these is new in this context. Lazy sweeping has been in common use for a decade. It ...

13 Software prefetching for mark-sweep garbage collection: hardware analysis and software redesign

Chen-Yong Cher, Antony L. Hosking, T. N. Vijaykumar

October 2004 **ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , ACM SIGARCH Computer Architecture News , Proceedings of the 11th international conference on Architectural support for programming languages and operating systems ASPLOS-XI**, Volume 38 , 39 , 32 Issue 5 , 11 , 5

Publisher: ACM Press

Full text available: pdf(165.32 KB) Additional Information: full citation, abstract, references, index terms

Tracing garbage collectors traverse references from live program variables, transitively tracing out the closure of live objects. Memory accesses incurred during tracing are essentially random: a given object may contain references to any other object. Since application heaps are typically much larger than hardware caches, tracing results in many cache misses. Technology trends will make cache misses more important, so tracing is a prime target for prefetching.Simulation of Java benchmarks runni ...

Keywords: breadth-first, buffered prefetch, cache architecture, depth-first, garbage collection, mark-sweep, prefetch-on-grey, prefetching

14 Profile-guided proactive garbage collection for locality optimization

Wen-ke Chen, Sanjay Bhansali, Trishul Chilimbi, Xiaofeng Gao, Weihaw Chuang

June 2006 **ACM SIGPLAN Notices , Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation PLDI '06**, Volume 41 Issue 6

Publisher: ACM Press

Full text available: pdf(406.15 KB) Additional Information: full citation, abstract, references, index terms

Many applications written in garbage collected languages have large dynamic working sets and poor data locality. We present a new system for continuously improving program data locality at run time with low overhead. Our system proactively reorganizes the heap by leveraging the garbage collector and uses profile information collected through a low-overhead mechanism to guide the reorganization at run time. The key contributions include making a case that garbage collection should be viewed as a ...

Keywords: cache optimization, data locality, garbage collectors, memory optimization, page optimization

15 Error-free garbage collection traces: how to cheat and not get caught

Matthew Hertz, Stephen M Blackburn, J Eliot B Moss, Kathryn S. McKinley, Darko Stefanović

June 2002 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '02**, Volume 30 Issue 1

Publisher: ACM Press

Full text available: pdf(105.06 KB) Additional Information: full citation, abstract, references, citings

Programmers are writing a large and rapidly growing number of programs in object-oriented languages such as Java that require garbage collection (GC). To explore the design and evaluation of GC algorithms quickly, researchers are using simulation based on traces of object allocation and lifetime behavior. The *brute force* method generates perfect traces using a whole-heap GC at every potential GC point in the program. Because this process is prohibitively expensive, researchers often use < ...

16 Memory system behavior of Java programs: methodology and analysis

Jin-Soo Kim, Yarsun Hsu

June 2000 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '00**, Volume 28 Issue 1

Publisher: ACM Press

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper studies the memory system behavior of Java programs by analyzing memory reference traces of several SPECjvm98 applications running with a Just-In-Time (JIT) compiler. Trace information is collected by an exception-based tracing tool called JTRACE, without any instrumentation to the Java programs or the JIT compiler.First, we find that the overall cache miss ratio is increased due to garbage collection, which suffers from higher cache misses compared to the application. ...

17 Some issues and strategies in heap management and memory hierarchies

Paul R. Wilson

January 1991 **ACM SIGPLAN Notices**, Volume 26 Issue 3

Publisher: ACM Press

Full text available: pdf(802.62 KB) Additional Information: full citation, citings, index terms

18 Mobile applications: Storing a persistent transactional object heap on flash memory

Michal Spivak, Sivan Toledo

June 2006 **Proceedings of the 2006 ACM SIGPLAN/SIGBED conference on Language, compilers and tool support for embedded systems LCTES '06**

Publisher: ACM Press

Full text available: pdf(337.46 KB) Additional Information: full citation, abstract, references, index terms

We present the design and implementation of TinyStore, a persistent, transactional, garbage-collected memory-management system, designed to be called from the Java virtual machine of a Java Card. The system is designed for flash-based implementations of Java Card, a variant of the Java platform for smart cards. In the Java Card platform, objects are persistent by default. The platform supports transactions: a sequence of accesses to objects can be explicitly declared to constit ...

Keywords: nor flash, Java Card, flash, persistent heaps, persistent object stores, smart cards, transactions

19 The design, implementation, and evaluation of adaptive code unloading for resource-constrained devices

Lingli Zhang, Chandra Krintz

June 2005 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 2 Issue 2

Publisher: ACM Press

Full text available: pdf(814.17 KB) Additional Information: full citation, abstract, references, index terms

Java Virtual Machines (JVMs) for resource-constrained devices, e.g., hand-helds and cell phones, commonly employ interpretation for program translation. However, compilers are able to produce significantly better code quality, and, hence, use device resources more efficiently than interpreters, since compilers can consider large sections of code concurrently and exploit optimization opportunities. Moreover, compilation-based systems store code for reuse by future invocations obviating the redund ...

Keywords: Code unloading, JIT, JVM, code-size reduction, resource-constrained devices

20 Sensor networks and performance analysis: Impact of virtual execution environments on processor energy consumption and hardware adaptation

Shiwen Hu, Lizy K. John

June 2006 **Proceedings of the 2nd international conference on Virtual execution environments VEE '06**

Publisher: ACM Press

Full text available: pdf(306.86 KB) Additional Information: full citation, abstract, references, index terms

During recent years, microprocessor energy consumption has been surging and efforts to reduce power and energy have received a lot of attention. At the same time, virtual execution environments (VEEs), such as Java virtual machines, have grown in popularity. Hence, it is important to evaluate the impact of virtual execution environments on microprocessor energy consumption. This paper characterizes the energy and power impact of two important components of VEEs, Just-in-time(JIT) optimization an ...

Keywords: energy efficiency, hardware adaptation, power dissipation

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	24768880	@ad<"20031010"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 13:38
L2	8	heap same cache same increas$6 same decreas$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 13:54
L3	4	1 and 2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 13:54
L4	17	heap same cache same dynamic$4 same siz$7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 13:54
L5	13	1 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:13
L6	2	"20030200392".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:13
L7	16390	heap$6	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:13
L8	1	6 and 7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:28
L9	4607	garbag$5 near2 collection$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:28

EAST Search History

L10	1282	cache near3 evict$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:29
L11	112266	allocation or deallocat$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:29
L12	48	9 and 10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:29
L13	24	12 and 11	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:29
L14	2	13 and 5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/07/29 14:29